UNIVERSIDAD NACIONAL DE RÍO CUARTO

FACULTAD DE CIENCIAS EXACTAS, FÍSICO-QUÍMICAS Y NATURALES

DEPARTAMENTO DE COMPUTACIÓN

CARRERA/S: Analista en Computación, Profesorado en Ciencias de la Computación, Licenciatura en Ciencias de la Computación

PLAN DE ESTUDIOS: 1999

ASIGNATURA: Diseño de Algoritmos y Algoritmos II **CÓDIGO: 3326/3302**

DOCENTE RESPONSABLE: Dr. Nazareno Aguirre

EQUIPO DOCENTE: Dr Nazareno Aguirre, Lic. Sonia Permigiani, Lic. Gastón Scilingo, Lic. Simón Gutiérrez.

AÑO ACADÉMICO: 2016

REGIMEN DE LA ASIGNATURA: Cuatrimestral

RÉGIMEN DE CORRELATIVIDADES:

Diseño de Algoritmos, para cursar y rendir respectivamente

Aprobada	*Regular*
–	Estructuras de Datos y Algoritmos (3325)

Aprobada
Estructuras de Datos y Algoritmos (3325)

Algoritmos II

Aprobada	*Regular*
--	Algoritmos I (3301)

Aprobada
Algoritmos I (3301)

CARGA HORARIA TOTAL: 112 horas semanales

TEÓRICAS: 56 hs **PRÁCTICAS:** 56 hs **LABORATORIO:**

CARÁCTER DE LA ASIGNATURA: Obligatoria, en ambos casos

A. CONTEXTUALIZACIÓN DE LA ASIGNATURA

Diseño de Algoritmos y Algoritmos II son asignaturas obligatorias de los planes de estudios de las carreras de Analista en Computación y Licenciatura en Ciencias de la Computación, respectivamente. Estas asignaturas, que se dictan en el tercer año de las respectivas carreras, constituyen la última materia del trayecto de algorítmica en la carrera, que es el núcleo central de las carreras de computación en la UNRC. Estas materias dan un cierre al trayecto, completando así los conocimientos fundamentales de algorítmica: elementos básicos de la programación y la algorítmica (Intr. a la Algorítmica y Programación), técnicas de programación más avanzadas y herramientas para consturir programas correctos (Programación Avanzada), la representación de datos en la programación (Estructuras de Datos y Algoritmos, Algoritmos I), y técnicas de diseño de algoritmos (Diseño de Algoritmos/Algoritmos II).

B. OBJETIVOS PROPUESTOS

El objetivo esencial de las materias es lograr que los alumnos se familiaricen con las técnicas de diseño de algoritmos más importantes, que sepan aplicarlas en la práctica, aprovechando los mecanismos disponibles para su implementación en lenguajes de programación particulares. Se espera sin embargo que los alumnos aprendan las técnicas de diseño de algoritmos fundamentales de manera independiente de lenguajes. Se espera además que los alumnos conozcan las diferentes clases de problemas, de acuerdo a su complejidad.

Un objetivo particular de estas materias es conseguir integrar los conocimientos obtenidos por los alumnos en todas las materias anteriores del área de algoritmos (Introducción a la Algorítmica y Programación, Programación Avanzada, Estructuras de Datos y Algoritmos, Algoritmos I). En esta materia ser hará un fuerte énfasis en el uso adecuado de tipos abstractos de datos y su definición, y de anotaciones de programas tales como invariantes y contratos en términos de pre- y post-condición. Se utilizarán también las estructuras de datos más importantes, vistas en materias anteriores del área de algoritmos.

C. CONTENIDOS BÁSICOS DEL PROGRAMA A DESARROLLAR

Revisión de conceptos previos. Anotaciones en programas. La importancia de las especificaciones. Pre- y post-condición. Invariantes de ciclos. Funciones variantes. Pruebas de corrección de programas. Técnicas de transformación de programas. Pruebas de terminación de programas. Estructuras de datos fundamentales. Tipos abstractos de datos y su implementación. Análisis de algoritmos. Concepto de tiempo de ejecución. Reglas de análisis de programas imperativos y recursivos.

Introducción a la resolución algorítmica de problemas. Técnicas básicas de diseño de algoritmos:

Fuerza bruta. Algoritmos basados directamente en definiciones de problemas. Búsqueda exhaustiva. Limitaciones de la técnica.

Divide & Conquer. Relación entre Divide & Conquer y la recursión. Técnicas más específicas basadas en Divide & Conquer: Decrease & Conquer y Transform & Conquer. Algunos algoritmos importantes basados en estas estrategias. Aplicación de la técnica en la práctica.

Técnicas avanzadas de diseño de algoritmos:

Programación Dinámica. La programación dinámica como un optimización a soluciones divide & conquer. Algunos algoritmos importantes que siguen esta estrategia. Aplicación de la técnica en la práctica.

Greedy. Algunos ejemplos de greedy como una técnica completa. Greedy como una estrategia de aproximación a soluciones. Algunos algoritmos importantes que siguen esta estrategia. Aplicación de la técnica en la práctica.

Backtracking. Backtracking y su relación con la búsqueda exhaustiva. Búsqueda local. Branch & Bound. Backtracking en problemas de búsqueda convencionales. Búsqueda en problemas con adversarios. Su aplicación a juegos. Poda alfa-beta del árbol de juego. Aplicación de la técnica en la práctica.

Otras técnicas de diseño de algoritmos. Enfoques algorítmicos al aprendizaje. Introducción a las redes neuronales. La evolución como motivadora en la construcción de algoritmos: los algoritmos genéticos. Los algoritmos genéticos y su relación con búsqueda heurística.

Clases de problemas de acuerdo a su complejidad. La relación entre complejidad espacial y temporal. P y NP. NP-Completeness. Problemas NP-hard. Ejemplos. Soluciones mediante aproximación para problemas NP-hard.

La importancia de las especificaciones de software. Especificaciones de requisitos y especificaciones de diseño. Lenguajes informales vs. lenguajes formales de especificación. La lógica en los lenguajes formales de especificaciones. Limitaciones en poder expresivo. Análisis de especificaciones mediante deducción. Ventajas y desventajas de la deducción como mecanismo de análisis. Técnicas de análisis automático. Limitaciones de las técnicas de análisis automático.

D. FUNDAMENTACIÓN DE LOS CONTENIDOS

Las técnicas de diseño de algoritmos constituyen herramientas fundamentales para la programación. Éstas ofrecen estrategias para, dada la descripción de un problema, construir soluciones algorítmicas para el mismo. Diferentes técnicas de diseño dan lugar a diferentes tipos de algoritmos, muchas veces con características de desempeño temporal o espacial muy diferentes. Conocer estas técnicas es de fundamental importancia en la formación de todo graduado en Ciencias de la Computación.

Debido a que las técnicas de diseño de algoritmos dan lugar a soluciones alternativas a problemas computacionales, se hace necesario poder comparar adecuadamente estas soluciones alternativas. Por esta razón, y basándose en los elementos básicos de análisis de tiempo de ejecución cubiertos en Estructuras de Datos y Algoritmos/Algoritmos I, en esta materia se estudian también las clases de problemas de acuerdo a su complejidad y se pone un énfasis particular en análisis de algoritmos de acuerdo a su complejidad tanto temporal como espacial.

Finalmente, y dado que los temas cubiertos son sumamente prácticos, las asignaturas tienen un énfasis particular en el desarrollo de soluciones algorítmicas para problemas concretos, y una fuerte práctica de programación.

E. ACTIVIDADES A DESARROLLAR

Como una forma de simplificar las tareas de los alumnos, se elegirán como lenguajes para la implementación de algoritmos el lenguaje funcional Haskell y el lenguaje orientado a objetos Java. Los alumnos ya tienen experiencia en estos lenguajes, adquiridas en materias anteriores. Con el fin de afianzar las técnicas de diseño de algoritmos estudiadas en la asignatura, se pedirá en algunos casos que se implementen soluciones en dos paradigmas diferentes, el funcional y el imperativo/orientado a objetos. Además, al abarcar la técnica de diseño *backtracking*, se experimentará con el uso de un tercer lenguaje de programación, *Prolog*.

Los dos trabajos prácticos obligatorios, cuya aprobación es requisito para la regularidad, tienen por objetivo lograr que los alumnos puedan aplicar la teoría aprendida en la resolución algorítmica de problemas mediante la aplicación de las distintas técnicas de diseño de algoritmos estudiadas. Además, permiten integrar los contenidos de esta asignatura a muchos otros aprendidos en otras materias, principalmente de programación y lenguajes (evaluando las diferencias entre los distintos paradigmas de programación, por ejemplo), y afianzar las capacidades adquiridas en las materias de lógica y matemática discreta. Se intentará utilizar ejemplos y problemas interesantes, en los cuales las soluciones más adecuadas sean difíciles de reconocer, intentando estimular al alumnado.

Se fomentará la lectura de material adicional y la auto organización de los alumnos en sus actividades. Además, se dejará en manos de los alumnos la instalación y manejo de las herramientas de software utilizadas en la asignatura, como una manera de estimular la práctica en cuestiones más técnicas (no necesariamente ligadas a los tópicos que la asignatura abarca), y la experiencia en la utilización de herramientas nuevas.

CLASES TEÓRICAS: 4 horas semanales

CLASES PRÁCTICAS: 4 horas semanales

CLASES DE TRABAJOS PRÁCTICOS DE LABORATORIO: Todas las clases prácticas serán de laboratorio.

F. NÓMINA DE TRABAJOS PRÁCTICOS

La materia contará con dos trabajos prácticos obligatorios, sin recuperación, que serán evaluados por los docentes de la materia y su aprobación es condición para la regularidad. El plazo para la resolución de cada uno de los trabajos prácticos es de una semana.

Además, se proveerá una serie de ejercicios adicionales (cuya resolución es opcional) que acompañarán cada una de las clases teórico-prácticas.

El objetivo de los trabajos prácticos obligatorios (cuya resolución deberá será en principio individual) es poder evaluar la aplicación de las técnicas aprendidas en situaciones concretas de tamaño mediano, y la correcta comprensión de los fundamentos teóricos subyacentes a las técnicas estudiadas. Permiten también detectar y corregir problemas, y así afianzar los conocimientos adquiridos en materias anteriores, de las áreas de matemática discreta, lógica y programación.

G. HORARIOS DE CLASES:

El dictado de la asignatura estará compuesto por cuatro horas semanales de clases teóricas, más cuatro horas semanales de clases prácticas, divididas en 2 comisiones, con horarios por la mañana y por la tarde.

Clases teóricas:
- jueves de 14 a 16, aula a confirmar
- viernes de 14 a 16, aula a confirmar

Clases prácticas (comisión de la mañana):
- lunes de 10 a 12, aula a confirmar
- jueves de 8 a 10, aula a confirmar

Clases prácticas (comisión de la tarde):
- lunes de 14 a 16, aula a confirmar
- viernes de 16 a 18, aula a confirmar

HORARIO DE CLASES DE CONSULTAS: Las clases de consulta se darán semanalmente bajo demanda de los alumnos, en horario a convenir con los mismos.

H. MODALIDAD DE EVALUACIÓN:

- **Evaluaciones Parciales:** La materia tendrá dos exámenes parciales, cada uno con una recuperación. Los exámenes parciales serán teórico-prácticos, y abarcarán la primera y la segunda mitad de los contenidos de la asignatura, respectivamente. La evaluación del alumnado será complementada con trabajos prácticos obligatorios (ver más abajo).

- **Evaluación Final:** El examen final para alumnos regulares se llevará a cabo mediante evaluación oral, si el número de alumnos evaluados así lo permite. Abarcará la totalidad de los contenidos de la asignatura, verificando que los alumnos hayan adquirido los conocimientos teóricos y puedan aplicarlos en casos concretos en la práctica.

Por otra parte, el examen final para alumnos libres estará compuesto por una primera instancia escrita, en la cual se evaluarán las capacidades de los alumnos para diseñar algoritmos utilizando las técnicas de diseño de algoritmos aprendidas en la asignatura. La segunda instancia del examen final para alumnos libres será similar al examen final para alumnos regulares.

- **CONDICIONES DE REGULARIDAD:** Las condiciones para la regularidad de la asignatura son las siguientes: Aprobación de dos exámenes parciales (nota mayor o igual a 5) y dos trabajos prácticos obligatorios (nota mayor o igual a 5).

- **CONDICIONES DE PROMOCIÓN:** Las condiciones son las mismas que para la regularidad, con 7 como nota promedio en los exámenes parciales. La nota de promoción se calculará como un promedio ponderado de las notas de trabajos prácticos y parciales.

PROGRAMA ANALÍTICO

A. CONTENIDOS

Revisión de conceptos previos. Anotaciones en programas. La importancia de las especificaciones. Pre- y post-condición. Invariantes de ciclos. Funciones variantes. Pruebas de corrección de programas. Técnicas de transformación de programas. Pruebas de terminación de programas. Estructuras de datos fundamentales. Tipos abstractos de datos y su implementación. Análisis de algoritmos. Concepto de tiempo de ejecución. Reglas de análisis de programas imperativos y recursivos.

Introducción a la resolución algorítmica de problemas. Técnicas básicas de diseño de algoritmos:

Fuerza bruta. Algoritmos basados directamente en definiciones de problemas. Búsqueda exhaustiva. Limitaciones de la técnica.

Divide & Conquer. Relación entre Divide & Conquer y la recursión. Técnicas más específicas basadas en Divide & Conquer: Decrease & Conquer y Transform & Conquer. Algunos algoritmos importantes basados en estas estrategias. Aplicación de la técnica en la práctica.

Técnicas avanzadas de diseño de algoritmos:

Programación Dinámica. La programación dinámica como un optimización a soluciones divide & conquer. Algunos algoritmos importantes que siguen esta estrategia. Aplicación de la técnica en la práctica.

Greedy. Algunos ejemplos de greedy como una técnica completa. Greedy como una estrategia de aproximación a soluciones. Algunos algoritmos importantes que siguen esta estrategia. Aplicación de la técnica en la práctica.

Backtracking. Backtracking y su relación con la búsqueda exhaustiva. Búsqueda local. Branch & Bound. Backtracking en problemas de búsqueda convencionales. Búsqueda en problemas con adversarios. Su aplicación a juegos. Poda alfa-beta del árbol de juego. Aplicación de la técnica en la práctica.

Otras técnicas de diseño de algoritmos. Enfoques algorítmicos al aprendizaje. Introducción a las redes neuronales. La evolución como motivadora en la construcción de algoritmos: los algoritmos genéticos. Los algoritmos genéticos y su relación con búsqueda heurística.

Clases de problemas de acuerdo a su complejidad. La relación entre complejidad espacial y temporal. P y NP. NP-Completeness. Problemas NP-hard. Ejemplos. Soluciones mediante aproximación para problemas NP-hard.

La importancia de las especificaciones de software. Especificaciones de requisitos y especificaciones de diseño. Lenguajes informales vs. lenguajes formales de especificación. La lógica en los lenguajes formales de especificaciones. Limitaciones en poder expresivo. Análisis de especificaciones mediante deducción. Ventajas y desventajas de la deducción como mecanismo de análisis. Técnicas de análisis automático. Limitaciones de las técnicas de análisis automático.

B. CRONOGRAMA DE CLASES Y PARCIALES

Semana	Día/Fecha	Teóricos	Día/Fecha	Prácticos	Día/Fecha	Laboratorios	Parciales / Recuperatorios
1	14 al 18 de marzo	Introducción al Diseño de Algoritmos. Revisión de conceptos previos de algoritmos y representación de datos	14 al 18 de marzo	Repaso de conceptos previos de algoritmos y representación de datos		Repaso de conceptos previos de algoritmos y representación de datos	
2	21 al 25 de marzo	Técnica de Diseño Fuerza Bruta	21 al 25 de marzo	Práctico sobre fuerza bruta en la resolución de problemas computacionales		Práctico sobre fuerza bruta en la resolución de problemas computacionales	
3	28 de Marzo al 1 de Abril	Técnica de Diseño Divide & Conquer	28 de Marzo al 1 de Abril	Práctico sobre Divide & Conquer en la resolución de problemas computacionales		Práctico sobre Divide & Conquer en la resolución de problemas computacionales	
4	4 al 8 de abril		4 al 8 de abril	Práctico sobre Divide & Conquer en la resolución de problemas computacionales		Práctico sobre Divide & Conquer en la resolución de problemas computacionales	
5	11 al 15 de abril	Técnica de Diseño Programación Dinámica	11 al 15 de abril	Práctico sobre Programación Dinámica en la resolución de problemas computacionales		Práctico sobre Programación Dinámica en la resolución de problemas computacionales	
6	18 al 22 de abril	Técnica de Diseño Greedy	18 al 22 de abril	Práctico sobre Greedy en la		Práctico sobre Greedy en la resolución	

				resolución de problemas computacionales		de problemas computacionales	
7	25 de abril al 29 de abril	Técnica de Diseño Búsqueda	25 de abril al 29 de abril	Práctico sobre Búsqueda en la resolución de problemas computacionales		Práctico sobre Búsqueda en la resolución de problemas computacionales	29/4 Primer parcial
8	2 al 6 de mayo	Búsqueda Informada, Búsqueda en Problemas con adversarios	2 al 6 de mayo	Práctico sobre Búsqueda informada y adversaria en la resolución de problemas computacionales		Práctico sobre Búsqueda informada y adversaria en la resolución de problemas computacionales	6/5 Recuperación del primer parcial
9	9 al 13 de mayo	Técnicas de aprendizaje automático: redes neuronales	9 al 13 de mayo	Práctico sobre Redes Neuronales en la resolución de problemas computacionales		Práctico sobre Redes Neuronales en la resolución de problemas computacionales	
10	16 al 20 de mayo	Algoritmos Genéticos	16 al 20 de mayo	Práctico sobre Algoritmos Genéticos en la resolución de problemas computacionales		Práctico sobre Algoritmos Genéticos en la resolución de problemas computacionales	
11	23 al 27 de mayo	Revisión y comparación de técnicas estudiadas	23 al 27 de mayo	Práctico sobre Algoritmos Genéticos en la resolución de problemas computacionales		Práctico sobre Algoritmos Genéticos en la resolución de problemas computacionales	
12	30 de Mayo al 3 de junio	Clases de problemas de acuerdo a su	30 de mayo al 3 de junio	Práctico sobre Algoritmos Genéticos en la		Práctico sobre Algoritmos Genéticos en la resolución	3/6 segundo parcial

		complejidad		resolución de problemas computacionales		de problemas computacionales	
13	6 al 10 de junio	Introducción a los modelos formales de software	6 al 10 de junio	Práctica sobre modelado formal en Alloy		Práctica sobre modelado formal en Alloy	10/6 recuperación del segundo parcial
14	13 al 17 de junio	Análisis automático de modelos formales de software	13 al 17 de junio	Práctica sobre análisis automático de modelos Alloy		Práctica sobre análisis automático de modelos Alloy	

C. BIBLIOGRAFÍA

Obligatoria

A. Levitin, *"Introduction to the Design and Analysis of Algorithms"*, Addison-Wesley, 2002.
A. Aho, J. Hopcroft y J. Ullman, *"Data Structures and Algorithms"*, Addison-Wesley, 1983.
D. Jackson, *"Software Abstractions"*, MIT Press, 2006.
T. Cormen, C. Leiserson, R. Rivest y C. Stein, *"Introduction to Algorithms"*, Third Edition, MIT Press, 2009.

De consulta

P. Hudak, J. Peterson y J. Fasel, *"A Gentle Introduction to Haskell"*, 2000.
U. Manber, *"Introduction to Algorithms, A Creative Approach"*, Addison-Wesley, 1989.
S. Baase, *"Computer Algorithms, Introduction to Design and Analysis"*, second edition, Addison-Wesley, 1988.
R. Bird, *"Introduction to Functional Programming using Haskell"*, second edition, Prentice Hall, 1998.
B. Meyer, *"Object Oriented Software Construction"* second edition, Addison-Wesley, 2000.
B. Liskov, *"Program Development in Java, Abstraction, Specification and Object-Oriented Design",* Addison-Wesley, 2001.